# Wie modelliert man ein Problem?

*Constraint Programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it.*

[Eugene C. Freuder, *Constraints Journal*, 1997]

- Mehrere Modellierungen eines Problems
  - Unterschiedliche Variablen
  - Unterschiedliche Constraints
- Unterschiedliches Effizienzverhalten abhängig von
  - Anzahl der Variablen
  - Art der Constraints

# Ein einfaches Zuordnungsproblem

- 4 Arbeiter: $w_1, w_2, w_3, w_4$
- 4 Produkte: $p_1, p_2, p_3, p_4$
- Ein Arbeiter wird einem Produkt zugeordnet und vice versa.

Der Profit eines Arbeiters $w_i$, der an einem Produkt $p_j$ ist:

| | $p_1$ | $p_2$ | $p_3$ | $p_4$ |
|---|---|---|---|---|
| $w_1$ | 7 | 1 | 3 | 4 |
| $w_2$ | 8 | 2 | 5 | 1 |
| $w_3$ | 4 | 3 | 7 | 2 |
| $w_4$ | 3 | 1 | 6 | 3 |

Das Problem ist gelöst, wenn der gesamte Profit mindestens 19 ist.

# Modellierung durch Boolesche Variablen (I)

Operations Research-Ansatz: 16 Boolesche Variablen

$B_{ij}$: Arbeiter $i$ wird einem Produkt $j$ zugeordnet.

Modellierung mit linearen arithmetischen Constraints:

- Arbeiter $w_i$ wird einem einzigen Produkt zugeordnet:

$$B_{i1} + B_{i2} + B_{i3} + B_{i4} = 1$$

- Produkt $p_j$ wird einem einzigem Arbeiter zugeordnet:

$$B_{1j} + B_{2j} + B_{3j} + B_{4j} = 1$$

# Modellierung durch Boolesche Variablen (II)

- Gesamter Profit:

$$
\begin{aligned}
P \quad = \quad & 7 * B_{11} + B_{12} + 3 * B_{13} + 4 * B_{14} \\
& + 8 * B_{21} + 2 * B_{22} + 5 * B_{23} + B_{24} \\
& + 4 * B_{31} + 3 * B_{32} + 7 * B_{33} + 2 * B_{34} \\
& + 3 * B_{41} + B_{42} + 6 * B_{43} + 3 * B_{44}
\end{aligned}
$$

# Modellierung durch Boolesche Variablen: Programm (I)

```
assignment(List) :-
      List = [B11,B12,B13,B14,B21,B22,B23,B24,
              B31,B32,B33,B34,B41,B42,B43,B44],
      gen_domains(List,0..1),
      B11 + B12 + B13 + B14 #= 1,
      B21 + B22 + B23 + B24 #= 1,
      B31 + B32 + B33 + B34 #= 1,
      B41 + B42 + B43 + B44 #= 1,
      B11 + B21 + B31 + B41 #= 1,
      B12 + B22 + B32 + B42 #= 1,
      B13 + B23 + B33 + B43 #= 1,
      B14 + B24 + B34 + B44 #= 1,
      P #= 7 * B11 + B12 + 3 * B13 + 4 * B14
         + 8 * B21 + 2 * B22 + 5 * B23 + B24
         + 4 * B31 + 3 * B32 + 7 * B33 + 2 * B34
         + 3 * B41 + B42 + 6 * B43 + 3 * B44,
      P #>= 19,
      labeling([],List).
```

# Modellierung durch Boolesche Variablen: Programm (II)

```
:- assignment([B11,B12,B13,B14,B21,B22,...]).
B11 = 0,   B11 = 0,   B11 = 0,   B11 = 1
B12 = 0,   B12 = 0,   B12 = 1,   B12 = 0
B13 = 0,   B13 = 0,   B13 = 0,   B13 = 0
B14 = 1,   B14 = 1,   B14 = 0,   B14 = 0
B21 = 1,   B21 = 1,   B21 = 1,   B21 = 0
B22 = 0,   B22 = 0,   B22 = 0,   B22 = 1
B23 = 0,   B23 = 0,   B23 = 0,   B23 = 0
B24 = 0,   B24 = 0,   B24 = 0,   B24 = 0
B31 = 0,   B31 = 0,   B31 = 0,   B31 = 0
B32 = 0,   B32 = 1,   B32 = 0,   B32 = 0
B33 = 1,   B33 = 0,   B33 = 1,   B33 = 1
B34 = 0,   B34 = 0,   B34 = 0,   B34 = 0
B41 = 0,   B41 = 0,   B41 = 0,   B41 = 0
B42 = 1,   B42 = 0,   B42 = 0,   B42 = 0
B43 = 0,   B43 = 1,   B43 = 0,   B43 = 0
B44 = 0,   B44 = 0,   B44 = 1,   B44 = 1
```

**Insgesamt 28 Ableitungsschritte**

# Modellierung durch Variablen für die Arbeiter (I)

4 Variablen: $W_1, W_2, W_3, W_4$

Wenn $i$-ter Arbeiter Produkt $j$ bearbeitet, dann hat $W_i$ den Wert $j$.

**Jede Variable hat einen unterschiedlichen Wert:**

$W_i \neq Wj$ für alle $1 \leq i \neq j \leq 4$

**Modellierung** durch das Constraint `all_distinct`

```
all_distinct([W1,W2,W3,W4])
```

# Modellierung durch Variablen für die Arbeiter (II)

## Gesmater Profit

Array je Arbeiter:

- $\text{profit}_i$: Profit eines Arbeiters $i$ für jedes Produkt
- $\text{profit}_i[j]$: Profit, wenn das Produkt $j$ von dem Arbeiter $i$ bearbeitet wird

  $\text{profit}_1[1] = 7$, $\text{profit}_1[2] = 1$, $\text{profit}_1[3] = 3$, $\text{profit}_1[4] = 4$

Der Profit des Arbeiters $i$: $\text{profit}_i[W_i]$

**Modellierung** durch das Constraint `element`

```
element(W1,[7,1,3,4],WP1)
```

## Das Constraint `element`

`element(?X,+List,?Y)`

- X und Y sind Zahlen- oder Domainvariablen
- `List` ist eine Liste von Zahlen oder Domainvariablen ist.
- `element(?X,+List,?Y)` gilt, wenn `Y` identisch mit dem `X`'ten Element der Liste `List` ist.
- Operational, sind die Bereiche von `X` und `Y` so constrained, daß für jedes Element aus dem Bereich von `X` ein kompatibles Element aus dem Bereich von `Y` gibt und vice versa.

```
:- Wert #> 5, element(N,[7,8,4,3],Wert).
Wert in 7..8, N in 1..2
```

# Modellierung durch Variablen für die Arbeiter: Programm (I)

```
assignment(W1,W2,W3,W4) :-
        W1 in 1..4,
        W2 in 1..4,
        W3 in 1..4,
        W4 in 1..4,
        all_distinct([W1,W2,W3,W4]),
        element(W1,[7,1,3,4],WP1),
        element(W2,[8,2,5,1],WP2),
        element(W3,[4,3,7,2],WP3),
        element(W4,[3,1,6,3],WP4),
        P #= WP1 + WP2 + WP3 + WP4,
        P #>= 19,
        labeling([],[W1,W2,W3,W4]).
```

# Modellierung durch Variablen für die Arbeiter: Programm (II)

```
:- assignment(W1,W2,W3,W4).

W1 = 1,     W1 = 2,     W1 = 4,     W1 = 4
W2 = 2,     W2 = 1,     W2 = 1,     W2 = 1
W3 = 3,     W3 = 3,     W3 = 2,     W3 = 3
W4 = 4,     W4 = 4,     W4 = 3,     W4 = 2
```

**Insgesamt 14 Ableitungsschritte**

# Modellierung durch Variablen für die Produkte: Programm (I)

```
assignment(T1,T2,T3,T4) :-
        T1 in 1..4,
        T2 in 1..4,
        T3 in 1..4,
        T4 in 1..4,
        all_distinct([T1,T2,T3,T4]),
        element(T1,[7,8,4,3],TP1),
        element(T2,[1,2,3,1],TP2),
        element(T3,[3,5,7,6],TP3),
        element(T4,[4,1,2,3],TP4),
        P #= TP1 + TP2 + TP3 + TP4,
        P #>= 19,
        labeling([],[T1,T2,T3,T4]).
```

# Modellierung durch Variablen für die Produkte: Programm (II)

```
:- assignment(T1,T2,T3,T4).

T1 = 1,    T1 = 2,    T1 = 2,    T1 = 2
T2 = 2,    T2 = 1,    T2 = 3,    T2 = 4
T3 = 3,    T3 = 3,    T3 = 4,    T3 = 3
T4 = 4,    T4 = 4,    T4 = 1,    T4 = 1
```

**Insgesamt 7 Ableitungsschritte**

**Grund:** Besseres Propagationsverhalten

`TP1 in 3..8, TP2 in 1..3, TP3 in 3..7, TP4 in 1..4`

Aus

$$TP_1 \geq min(P) - max(TP_2) - max(TP_3) - max(TP_4)$$

folgt $TP_1 \geq 5$. Das Constraint `element(T1,[7,8,4,3],TP1)` reduziert dann den Bereich von `T1` (`T1 in 1..2`).

# Vergleich der Modellierungen (I)

Eine schwierige Aufgabe

## Einige Kriterien

- Effizienz
  - Propagation der Constraints
  - Anzahl der Variablen (i.a. je weniger, desto besser)
- Flexibilität: Zusätzliche Constraints
  - Abhängig von der Art der Constraints

# Vergleich der Modellierungen (II)

- Verbiete, daß gleichzeitig Arbeiter 1 Produkt 1 zugeordnet wird und Arbeiter 4 Produkt 4.
  In der booleschen Modellierung einfach auszudrücken: $B_{11} + B_{44} \leq 1$. In der zweiten und dritten Modellierung schwieriger!

- Arbeiter 3 wird einem Produkt zugeordnet, dessen Nummer größer ist, als die Nummer des Produkts, das vom Arbeiter 2 durchgeführt wird. Im zweiten Modell einfach auszudrücken: $W_3 > W_2$. In der booleschen Modellierung schwieriger! In der dritten Modellierung noch schwieriger!!

**Ausweg:**

Kombination unterschiedlicher Modellierungen: Redundante Constraints